CPE Lyon – 4IRC – Année 2025/26
Projet Cellular IoT

Nikolaï LEBEDEV, Antoine BONNEAU, Oscar CARRILLO

Introduction générale

Contexte et motivation

Les systèmes IoT modernes combinent des réseaux locaux contraints, des capacités de *traitement en périphérie (edge computing)*, et des *connexions longue portée vers le cloud*. Ces systèmes doivent fonctionner avec des ressources limitées, tout en garantissant fiabilité, sécurité et exploitabilité des données.
Ce projet s'inscrit dans ce contexte et vise à confronter les étudiants à la conception et à l'implémentation d'une **chaîne IoT complète**, depuis la collecte des données sur le terrain jusqu'à leur exploitation dans le cloud. L'accent est mis sur des problématiques concrètes rencontrées dans des déploiements IoT réels : contraintes réseau, gestion des flux, résilience, sécurité des communications et traitement local des données.

Objectifs pédagogiques généraux

À l'issue du projet, les étudiants devront être capables de :
- Concevoir et implémenter une architecture IoT distribuée intégrant capteurs, traitement local et cloud
- Mettre en œuvre un protocole applicatif adapté à des systèmes embarqués contraints
- Intégrer des capteurs physiques et/ou des modules de vision dans une application fonctionnelle
- Concevoir un traitement local (edge) permettant de filtrer, agréger ou détecter des événements
- Analyser les performances du système (latence, pertes, charge, robustesse)
- Appliquer des mécanismes de sécurité de base (authentification, intégrité, anti-rejeu)

Objectifs spécifiques selon le parcours

Selon le sujet attribué, certains objectifs techniques spécifiques peuvent s'ajouter :
- **Parcours capteurs / communication IoT** (Figure 1) :
  - Mise en œuvre d'un réseau local basse consommation basé sur **802.15.4** (topologie étoile ou mesh selon le sujet)
  - Analyse des contraintes liées aux communications locales (débit, latence, pertes, robustesse)
- **Parcours vision / traitement embarqué** (Figure 2) :
  - Mise en œuvre d'un **traitement d'image local** sur système embarqué contraint
  - Conception d'une approche **événementielle** limitant les volumes de données transmis
  - Prise en compte de principes de **privacy-by-design**

Description générale du travail

Le système étudié est composé de plusieurs nœuds IoT déployés sur le terrain et interconnectés par un réseau local basse consommation lorsque nécessaire. Selon le sujet, ces nœuds assurent la collecte de données capteurs ou l'analyse locale d'images.
Les données produites sont transmises vers une **passerelle edge**, chargée d'agréger les flux, d'appliquer un traitement local (filtrage, détection d'événements ou logique décisionnelle), et de sélectionner les informations pertinentes à transmettre vers le cloud.

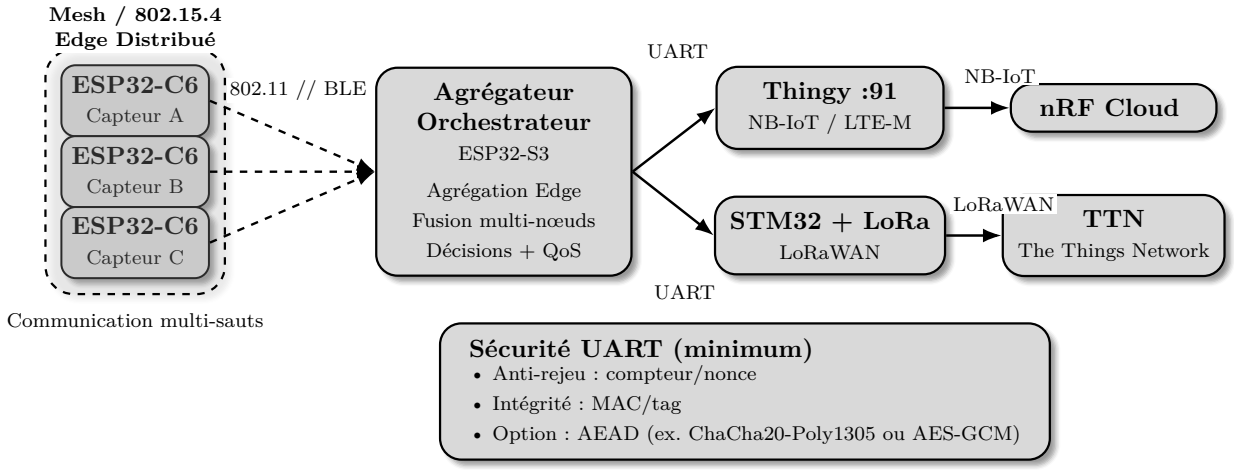

FIGURE 1 – Architecture du parcours capteurs : réseau IoT (ESP32-C6) avec agrégation sur passerelle (ESP32-S3) et remontée vers le cloud via NB-IoT ou LoRaWAN.

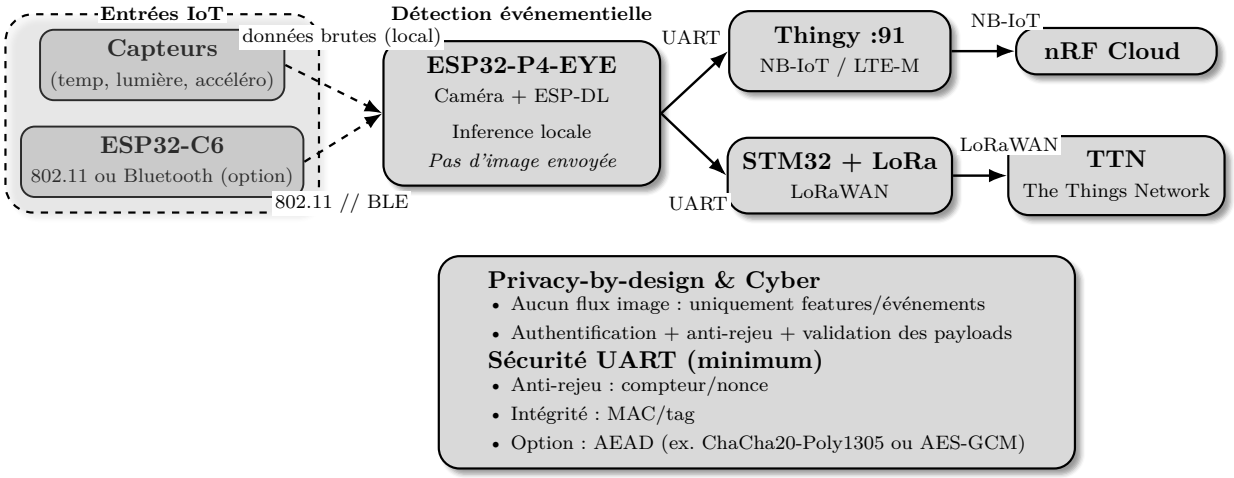

FIGURE 2 – Architecture du parcours vision : analyse d'image locale sur ESP32-P4-EYE, enrichie par des capteurs, avec envoi d'événements vers le cloud.

La communication entre la passerelle et l'infrastructure d'uplink est réalisée via une liaison série (**UART**), pouvant être sécurisée. Le cloud est utilisé pour l'**exploitation des données** (visualisation, alertes, supervision), et non comme un simple point de calcul central (Figure 3).

Architecture attendue

L'architecture générale des projets repose sur les éléments suivants :

- **Nœuds terrain :** ESP32-C6 (capteurs, communication 802.15.4 si utilisée)
- **Passerelle edge :** ESP32-S3 (parcours capteurs) ou ESP32-P4 (parcours vision)
- **Uplink longue portée :** LoRaWAN ou NB-IoT
- **Plateforme cloud :** The Things Network (TTN) ou nRF Cloud, avec une chaîne de traitement cloud commune (Figure 3)

Organisation des sujets

Les projets sont organisés autour de deux parcours techniques :

- un parcours orienté **capteurs et communication IoT**,
- un parcours orienté **vision et traitement embarqué**.

Chaque sujet est décliné en deux variantes technologiques, selon la solution d'uplink retenue :

- **Variante NB-IoT** (Thingy :91 et nRF Cloud),

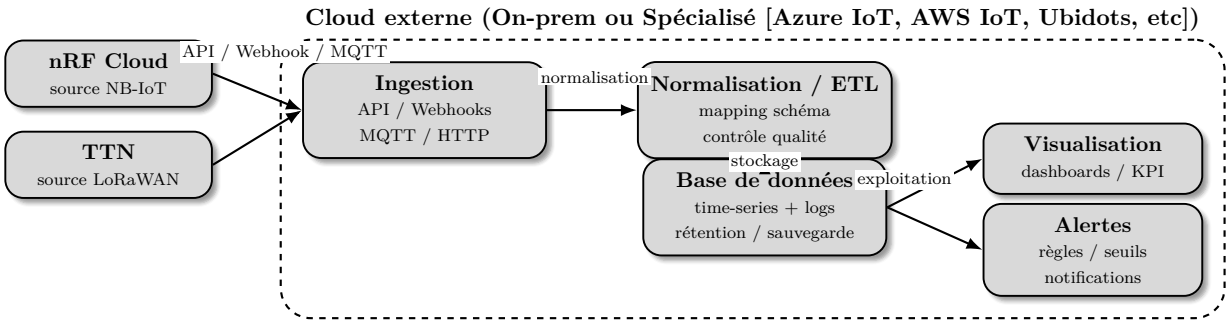

FIGURE 3 – Chaîne de traitement cloud : récupération des données depuis TTN et/ou nRF Cloud, ingestion, stockage, visualisation et alertes.

— **Variante LoRaWAN** (STM32 + LoRa et The Things Network).

La description détaillée des sujets S1, S2 et V1 est présentée dans les pages annexes à ce document.

Répartition des groupes et choix des sujets

La répartition des étudiants par groupe de projet est donnée dans le Tableau 2. Il y a **11 groupes** (G1 à G11).

Les sujets sont proposés sous forme de **slots** (Tableau 1) : un slot correspond à l'association d'un **sujet** (S1, S2 ou V1) et d'une **technologie d'uplink** (**NB-IoT** ou **LoRaWAN**). Au total, **12 slots** sont proposés pour **11 groupes** : un slot restera donc non attribué.

Chaque groupe (G1–G11) devra choisir un **slot** et en informer l'intervenant. L'attribution des slots se fera selon la règle **premier arrivé, premier servi**, sous réserve de validation et des contraintes matérielles.

TABLE 1 – Slots de projets proposés

Slot	Sujet	Parcours	Uplink
S1-A	S1 — QoS & télémétrie	Capteurs / Réseaux	NB-IoT
S1-B	S1 — QoS & télémétrie	Capteurs / Réseaux	LoRaWAN
S1-C	S1 — QoS & télémétrie	Capteurs / Réseaux	NB-IoT
S1-D	S1 — QoS & télémétrie	Capteurs / Réseaux	LoRaWAN
S2-A	S2 — Store & Forward	Capteurs / Réseaux	NB-IoT
S2-B	S2 — Store & Forward	Capteurs / Réseaux	LoRaWAN
S2-C	S2 — Store & Forward	Capteurs / Réseaux	NB-IoT
S2-D	S2 — Store & Forward	Capteurs / Réseaux	LoRaWAN
V1-A	V1 — Vision événementielle	Vision / IA embarquée	NB-IoT
V1-B	V1 — Vision événementielle	Vision / IA embarquée	LoRaWAN
V1-C	V1 — Vision événementielle	Vision / IA embarquée	NB-IoT
V1-D	V1 — Vision événementielle	Vision / IA embarquée	LoRaWAN

Cyber-sécurité — exigences communes (OWASP IoT)

Quel que soit le sujet traité, le projet devra intégrer des mécanismes de cybersécurité minimaux, inspirés des bonnes pratiques **OWASP IoT**, afin de protéger les communications, les données et le système contre des attaques simples mais réalistes.

Exigences obligatoires

Les mécanismes suivants doivent être implémentés et démontrés :

TABLE 2 – Répartition des étudiants par groupe de projet

Groupe	Étudiant
G1	Mohamed Ait El Hadj
	Najd Ismail
	Léo Pellissier
	Nathan Courty
G2	Adel Ayadi
	Rémi Jara
	Yvan Pignier
	Alexis Escobar
G3	Samuel Backy
	Baptiste Lamoine
	Alrik Probst
	Ronan Fourneuve
G4	David Belkacemi
	Maxence Lerda
	Arthur Remy
	Mikael Gagaa
G5	Elliot Casna
	Messaoud Louchene
	Julien Rissons
	Vincent Geay
G6	Cédric Darmaisin
	Martin Louedec
	Melih Saglik
	Dayane Mahathy
G7	Mylan Forest
	Théo Marzolf
	Hasnène Sheikh
	Angelo Naudts
G8	Charles Gauthier
	Rayane Merlin
	Imad Touil
	Ndeye Aissatou Ndao
G9	Ivanne Giusti
	Lyes Mesbahi
	Léon Vanden Borre
	Clément Radermakers
G10	Matéo Guenot
	Valentine Paul
	Evan Ryser
G11	Sacha Henry
	Alexis Pecheri
	Tristan Thibout

— **Authentification des messages** : chaque message échangé doit permettre de vérifier son origine (ex. identifiant de nœud + secret partagé).
— **Protection contre le rejeu** : utilisation d'un compteur monotone, d'un nonce ou d'un timestamp permettant de détecter et rejeter les messages rejoués.
— **Validation stricte des entrées** : vérification systématique des longueurs, types et bornes des champs des messages reçus, côté nœuds et/ou côté passerelle.
— **Journalisation des événements de sécurité** : génération de logs minimaux (ex. erreurs d'authentification, messages rejetés, anomalies détectées) exploitables pour le diagnostic.

Exigences recommandées

Les mécanismes suivants ne sont pas strictement obligatoires mais seront valorisés s'ils sont correctement implémentés et justifiés :

— **Sécurisation de la liaison UART** : protection de la communication passerelle–uplink par des mécanismes d'intégrité et d'anti-rejeu.
— **Chiffrement des échanges UART** : mise en œuvre d'un schéma *authenticated encryption* (ex. AES-GCM ou ChaCha20-Poly1305) assurant confidentialité et intégrité.
— **Gestion des secrets** : stockage minimal des clés/secrets (pas de clé en clair dans les logs ou le code), initialisation contrôlée.

Justification et démonstration

Les choix de sécurité devront être :

— **documentés** (menaces visées, hypothèses, limites),
— **justifiés** par rapport aux contraintes du système (ressources, réseau, latence),

— et, dans la mesure du possible, **démontrés** lors de la soutenance (ex. rejet d'un message rejoué ou malformé).

Critères de validation

L'évaluation du projet reposera notamment sur :

— La démonstration fonctionnelle du **système complet** (terrain → passerelle → cloud)
— La **justification des choix techniques** et des compromis réalisés

Livrables attendus

Chaque groupe devra fournir :

— Un **dépôt Git** contenant le code, un `README` et les scripts nécessaires pour recréer votre projet.
— Un **schéma d'architecture** adapté au sujet traité + diagrammes de séquences
— Une **démonstration** du système (live ou vidéo)
— Un **rapport technique** de 4 à 6 pages : design, choix QoS, protocole UART, résultats expérimentaux

Sujet S1 — Réseau local 802.15.4 : QoS et télémétrie vers le cloud

Parcours : Capteurs / Réseaux IoT (Thread 802.15.4)
Architecture : Figure 1 (terrain→uplink), Figure 3 (traitement cloud)
Objectif : Concevoir un réseau Thread multi-sauts et une passerelle ESP32-S3 capables de *prioriser* les messages (QoS) et de remonter une télémétrie exploitable vers le cloud via NB-IoT ou LoRaWAN.

Contexte

Dans de nombreux déploiements IoT (bâtiments intelligents, sites industriels, infrastructures distribuées), plusieurs capteurs partagent un réseau local basse consommation et doivent transmettre à la fois des données de télémétrie périodiques et des événements critiques. Ces flux n'ont pas la même importance et ne doivent pas être traités de manière identique, en particulier lorsque le réseau est soumis à des contraintes de débit ou de congestion.

Matériel

— 3–5 × ESP32-C6 (nœuds capteurs Thread) : luminosité, température/fièvre, pression, accéléromètre, RGB (selon dispo)
— 1 × ESP32-S3 (passerelle / agrégateur)
— **Variante A (NB-IoT)** : 1 × Thingy :91 (uplink nRF Cloud) via UART
— **Variante B (LoRaWAN)** : 1 × STM32+LoRa (uplink TTN) via UART

Travail attendu

1. **Réseau Thread :** mise en place du mesh (leader/router/end-device) et adressage IPv6.
2. **Protocole applicatif :** UDP ou CoAP avec au moins *2 classes de priorité* (ex : alarme vs télémétrie).
3. **QoS côté passerelle :** file(s) de messages, ordonnancement, et politique de rejet/compactage en cas de surcharge.
4. **Uplink UART :** définition d'un framing stable (longueur, type, checksum/MAC, ACK/timeout).
5. **Export cloud :** remontée des mesures (et métadonnées : timestamp, node-id, RSSI/LQI si dispo) vers nRF Cloud (A) ou TTN (B).
6. **Mesures :** latence bout-en-bout, taux de perte, débit utile, impact des priorités (cas normal vs cas de surcharge).

Sujet S2 — Résilience : Store & Forward

Parcours : Capteurs / Réseaux IoT
Architecture : Figure 1, Figure 3
Objectif : Garantir la *livraison* des données malgré des coupures (NB-IoT/LoRaWAN) en mettant en œuvre une stratégie Store & Forward sur l'ESP32-S3.

Contexte

Les technologies de communication longue portée comme NB-IoT ou LoRaWAN présentent des avantages importants en termes de couverture et de consommation, mais elles peuvent être sujettes à des indisponibilités temporaires, des latences variables ou des pertes de connectivité. Un système IoT robuste ne doit pas perdre de données lors de ces interruptions, et doit être capable de reprendre proprement la transmission une fois la connexion rétablie.

Matériel

— 3–5 × ESP32-C6 (capteurs) + 1 × ESP32-S3 (passerelle)
— **Variante A** : Thingy :91 (nRF Cloud) / **Variante B** : STM32+LoRa (TTN)

Travail attendu

1. **Collecte Thread :** acquisition capteurs et transport jusqu'à la passerelle.
2. **Bufferisation :** file persistante côté S3 (RAM, avec justification des limites et du dimensionnement).
3. **Reprise :** ACK end-to-end (S3 ↔ uplink) + retransmission + stratégie backoff.
4. **Gestion du backlog :** métriques (taille file, âge moyen, plus vieux message) + politique de purge.
5. **Expérimentation :** simuler des coupures uplink (désactivation, radio down) et prouver la reprise.

Démo

— Coupure uplink 2–5 minutes, backlog, puis rattrapage contrôlé

Sujet V1 — Vision événementielle et remontée d'alertes

Parcours : Vision / IA embarquée
Architecture : Figure 2 (vision), Figure 3 (traitement cloud)
Objectif : Réaliser une détection locale simple (classification/détection d'un événement) sur ESP32-P4-EYE et ne remonter au cloud que des *événements* (pas d'images).

Contexte

Dans de nombreux cas d'usage, l'envoi continu d'images ou de flux vidéo vers le cloud est incompatible avec les contraintes de bande passante, de consommation énergétique ou de protection de la vie privée. Une alternative consiste à réaliser l'analyse directement sur le dispositif embarqué et à ne transmettre que des événements ou des résultats synthétiques.

Matériel

— 1 × ESP32-P4-EYE (caméra)
— Option capteurs : 1–2 × ESP32-C6 (ex : luminosité, accéléromètre) pour enrichir la décision
— **Variante A** : Thingy :91 (nRF Cloud) / **Variante B** : STM32+LoRa (TTN)

Travail attendu

1. **Pipeline vision :** acquisition, pré-traitement, inference (ESP-DL) sur P4-EYE.
2. **Définition d'événement :** message compact (classe + score + timestamp + contexte).
3. **Gating réseau :** démontrer la réduction du volume de données transmises par rapport à une approche non événementielle.
4. **Mesures :** taux de faux positifs / faux négatifs sur un scénario simple (à définir et documenter).

Privacy & Cyber

— **Interdiction d'envoyer des images**

Démo

— Événement détecté et visible dans le cloud (TTN/nRF + Figure 3)